The Report Committee for Paul Henderson
Certifies that this is the approved version of the following report:

Voiceprint Vault: Voice Authentication Service

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Suzanne Barber

Adnan Aziz

Voiceprint Vault: Voice Authentication Service

by

Paul Martin Henderson, B.E.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2013

Dedication

This work is dedicated to my wife, Shawna, and our son, Wyatt. Shawna, your ceaseless love and enthusiastic spirit are a blessing to me every day. Wyatt, your smiles have been a constant source of joy to me in all that I do.

Acknowledgements

I am deeply grateful to Dr. Barber for supervising this report and providing opportunities to share my research. I am also indebted to Dr. Aziz for inspiring this project in his class and for all his feedback on this report. Without the time and encouragement to investigate the concept of a speaker identification framework, this project never would have come to fruition.

Abstract

Voiceprint Vault: Voice Authentication Service

Paul Martin Henderson, M.S.E.

The University of Texas at Austin, 2013

Supervisor: Suzanne Barber

In a world dominated by smartphones, cloud computing, and online accounts, security of personal and corporate data is a critical concern. Voiceprint Vault provides a voice authentication service that can be used in a multitude of applications to secure sensitive data. Voiceprint Vault includes the following high-level features:

- Cloud-based voice authentication using trusted signal processing algorithms
- Multifactor authentication with use of optional password
- Cross-platform compatibility using secure web requests to authenticate
- Built-in storage and synching of private user data
- Java library to facilitate integration with Android applications

The Voiceprint Vault service allows users of an application to create an account, provide a voice sample, and then access the account with a simple spoken phrase. When users access their account, their voice sample is analyzed and compared to their training recordings. This system can be tailored to the needs of a particular user with per-user security options. It provides the convenience of voice access, but also allows for a password to be used for increased security.

The Voiceprint Vault service is designed to allow application developers to integrate an existing, tested authentication system into their app rather than creating their own authentication system. The Voiceprint Vault server provides application specific repositories that developers can create to hold all user data, cryptographic information, and voice samples. The user data stored on the Voiceprint Vault server provides built-in synchronization across all connected devices.

A reference implementation is provided that demonstrates the use of Voiceprint Vault authentication. The reference implementation is an Android app that uses the voice authentication service to protect access to personal notes, tasks, and dates that are synched across devices. Detailed instructions for integrating Voiceprint Vault into an existing application are also provided with the reference implementation.

The accuracy of voiceprint authentication was investigated and optimized for a set of sample users and recordings. The security features and dangers of such a system are described along with recommendations for safe use. The optimal parameters to be used in the voice authentication algorithms are also presented in this report.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 User Authentication Problem

Cloud computing, smartphone usage, and social networking present a growing security threat to individuals and corporations as more and more sensitive data is stored on the Internet. A great majority of smartphone and tablet applications are targeted toward users who are ignorant of the security measures protecting their data and simply expect the provider of the application to guarantee their security. A few examples of sensitive data include that stored in banking applications, email clients, e-commerce sites, and corporate servers. This environment has led to climbing rates of compromised accounts, data theft, corporate espionage, and a need for better security measures around sensitive user data. [MCC07]

1.2 Multifactor Authentication Options

A commonly proposed solution to the growing vulnerability to data theft is to use multifactor authentication. Multifactor authentication better protects users by requiring them to identify themselves with more than one type of data. For example, a user may be required to enter a PIN and scan their fingerprint. By requiring multiple forms of identification, these systems are more difficult to penetrate by impostors.

The three categories of multifactor authentication are knowledge, possession, and inherence. Knowledge factors require the user to reveal some piece of information that only they should know. Common knowledge-based factors include passwords, PINs, and security answers. Possession factors require the user to carry some physical token with them that would be very difficult for an identity thief to obtain. Possession factors can

range from a smartphone or one-time password token to a credit card or badge. Inherence factors require a user to identify themselves with biometric data. Common inherence authentication systems use fingerprints, voiceprint, retinal/iris scans, or even facial recognition.

Users and organizations are often resistant to multifactor authentication (MFA) for a number of reasons. MFA systems require additional costs and are often less convenient from a user perspective. Knowledge factors are most commonly used because password authentication is relatively easy to implement, very cheap, and operates reliably. Inherence factors are growing in use, but are still not very prevalent due to excessive cost. Possession factors face the problem that most users do not want to be required to carry a physical token with them at all times in order to access their accounts.

1.3 VOICEPRINT AUTHENTICATION

Biometric authentication factors provide greater convenience for the user, because there is no knowledge or physical device that the users must carry in order to authenticate themselves. Of the possible inherence factors that can be used for authentication, voiceprint authentication stands out as a method that could be made available to the greatest number of users. For most biometric factors, custom hardware is required to capture the user's biometric data. This may include a fingerprint scanner, retinal/iris scanner, or facial recognition camera. Voiceprint capture requires only a microphone, and with over one billion smartphone users worldwide in 2012[YFN13], a voiceprint authentication system could be used by an incredible number of users.

Despite some great advantages of voiceprint authentication, this technology has not yet gained widespread use. While speech recognition has taken off in recent years, most notably Apple's Siri[SIR13] for iOS, speaker recognition has not experienced the

same growth. Voiceprint authentication relies on speaker recognition, which is distinctly different from speech recognition, in that speaker recognition attempts to identify unique qualities of a voice whereas speech recognition attempts to determine what words are being spoken. For the purposes of this report, the terms "speaker recognition" and "voiceprint authentication" will both indicate that the inherent qualities of a user's voice are being analyzed rather than the words spoken by a user.

1.4 PROJECT SCOPE

The primary goal of this project was to develop an authentication service that could be conveniently integrated into new and existing applications. The authentication system uses text-independent voice authentication, because this provides security and convenience for the user. The Voiceprint Vault authentication system also delivers the following features:

- Multifactor authentication is provided by way of an optional password
- User data is automatically stored in Google App Engine and shared across devices
- Authentication service is cloud-based to allow devices from all platforms to use this service
- A reference implementation of an application using Voiceprint Vault is provided as an Android app
- A Java library is included to assist developers in integrating Voiceprint Vault authentication into their application

This service is hosted on Google App Engine to allow for strong consistency and scalability. Hosting this authentication service in the cloud allows it to be integrated with any client application that has network access. This service uses proven signal processing

algorithms to extract features from an audio sample for verification purposes. It also uses strong cryptographic protocols to protect users' data and login information. The frameworks and libraries that this system is built on are described in more detail in Chapter 2.

This system uses a client-server architecture to provide an accessible and consistent interface across platforms. This approach also facilitates synching of user data and helps to reduce the amount of code that must be implemented for each compatible platform. The system architecture is discussed at length in Chapter 3.

Since this service can protect sensitive data across a variety of applications, the security features are critically important. All network communication is completed with secure web requests and all user accounts and access codes are protected with algorithms that make brute force attacks infeasible. The security features of Voiceprint Vault are covered in Chapter 4.

A reference implementation of an application using Voiceprint Vault authentication is provided to demonstrate various features of this system. The example is an Android application that grants secure access to personal notes, tasks, and data that are synched across all connected devices. This application is described in Chapter 5.

Chapter 6 details the steps that a developer must complete to integrate Voiceprint Vault authentication into a new or existing app. These steps are available to prospective clients of Voiceprint Vault along with a Java library to facilitate integration.

The authentication system is evaluated in Chapter 7. Given a set of sample users and voice recordings, the parameters used with the voice authentication algorithm are optimized. The accuracy of this authentication system is analyzed with respect to variations in the threshold setting that controls user verification. Future extensions to this

project are considered and the project is compared against existing voice authentication systems.

Chapter 2: Technology Stack Overview

This system relies on a number of frameworks, libraries, and algorithms to accomplish its high-level objectives. The Voiceprint Vault server is hosted on Google App Engine. The verification process uses Mel Frequency Cepstrum Coefficients to extract features from the voice. The service provides access tokens that are encrypted with the Advanced Encryption Standard. This chapter describes the dependent technologies of the Voiceprint Vault system.

2.1 GOOGLE APP ENGINE FRAMEWORK

This service may support a large number of potential clients and must provide synchronized data across many devices on the network. Devices across many mobile platforms should be able to communicate with the server by web requests. Google App Engine was chosen as the framework to host the Voiceprint Vault server because it solves these requirements and provides many additional features that simplify the server development. App Engine provides for high levels of consistency, availability, and scalability. With this framework, Voiceprint Vault is able to support an indefinite number of client applications without constantly monitoring and adding servers. App Engine includes the Datastore to hold user and application data and also supports Java servlets, which allow other dependent libraries to be conveniently integrated. [AEN13]

2.1.1 Datastore

A primary goal for this library and its surrounding framework is that user data stored on the Voiceprint Vault Server is immediately available on any Android device from which the user accesses their account. This requires that the server provides vast data storage that is highly available and consistent. With future extensions of this system

to allow shared group data, this could even require that users around the world are concurrently viewing and modifying shared data.

Google App Engine's Datastore provides an ideal solution to this requirement. Using the App Engine framework, the Voiceprint Vault Server is served by many servers located in multiple geographically-dispersed locations. A Voiceprint Vault client could be connecting to a different App Engine server with each successive request based on dynamic internet traffic, but the App Engine Datastore still provides consistent query results under a given entity. [GDS12]

2.1.2 Servlets

The App Engine framework allows Java Servlets to be used to handle and respond to web requests. Servlets are Java classes that run in response to web requests to dynamically handle the request and build a web page. Servlets allow App Engine projects to take advantage of many benefits of Java. In the case of Voiceprint Vault, this allows for the use of mature libraries and extensive code reuse. [ORA13]

When a HTTP request comes into the App Engine server, it is redirected to the appropriate servlet to handle the request. The web.xml deployment descriptor maps all public URLs to their associated servlets. The servlet class mapped by web.xml must extend the `HttpServlet` class and implement the `doGet` and/or `doPost` methods. These methods receive an `HttpServletRequest` object and an `HttpServletResponse` object. From these objects, parameters from a web request can easily be parsed and the resulting web page written. [GAE13]

2.2 VOICE PROCESSING ALGORITHMS

The voice processing algorithm is critical to the operation of this service because without successfully verifying a user's voice, the entire framework is rendered

ineffective. Several different algorithms for voice processing exist that are catered to different purposes. Different methods are used depending on whether the audio sample is text-dependent or text-independent and whether speaker verification or speaker identification is required. This application makes use of text-independent speaker verification in order to provide secure access to the user's account without requiring that the user speak a hidden password. This is more convenient for the user and enables the user to authenticate their account in front of others without giving away a secret passphrase.

At a high level, speaker verification requires a feature extraction phase and a feature comparison phase. In the feature extraction phase, the vocal sample is analyzed to produce a set of parameters that concisely represent the frequencies present in the sample. During the feature comparison phase, the parameters that represent the signal are compared against training recordings for the intended user. If the difference between the two sets of parameters falls below a predetermined acceptance threshold, then the user is successfully verified. If the difference is above the acceptance threshold, then the user is rejected as an invalid user.

The two most common algorithms for feature extraction used are Mel Frequency Cepstrum Coefficients (MFCC) and Linear Predictive Coding (LPC). The MFCC algorithm is the most popular algorithm for speaker recognition now in large part because it uses a logarithmic scale for the frequencies extracted in order to better model the human auditory response. [KAM10] This project uses the MFCC algorithm for feature extraction.

After the features have been extracted from an audio sample, speaker verification requires that these features be compared to a set of features computed from training recordings. Several algorithms exist for comparing feature sets, including Dynamic Time

Warping, Hidden Markov Modeling, and Vector Quantization. The Vector Quantization approach is most commonly used. It plots the training features on a two-dimensional graph and groups neighboring points into centroids. At speaker verification time, the user's extracted features are compared against their nearest centroids to determine an absolute difference from the training features. [RAS04] For this application, a simple averaged difference of the feature coefficients is used to compute the absolute difference. This approach was used for its quick implementation and positive experimental results. Future improvements to this algorithm are discussed in Chapter 8.

2.2.1 Mel Frequency Cepstrum Coefficients

The human voice is considered to be a quasi-stationary signal, because it is only slowly varying in time. Over a period of five ms to 100 ms, the characteristics typically do not change. [KAM10] For that reason, a signal is windowed into frames, ranging from 20 ms to 40 ms in length prior to spectral analysis. This allows the individual frames to represent a relatively static vocal sample, which is an important requirement for MFCC processing. This window is big enough to include enough samples for a reliable spectral analysis, but small enough that the signal within the window does not change excessively. [LYO09]

From this point, the discrete Fourier transform is used to transform the windowed signal into the frequency domain. The resulting spectrum is mapped onto the logarithmic Mel scale. [ZHE01] Above 1000 Hz, increasing pitches are more difficult for the human ear to differentiate. The Mel scale uses a logarithmic mapping of the raw pitch so that higher pitches that humans would place less importance on do not skew the analysis of this audio sample. [RAS04]

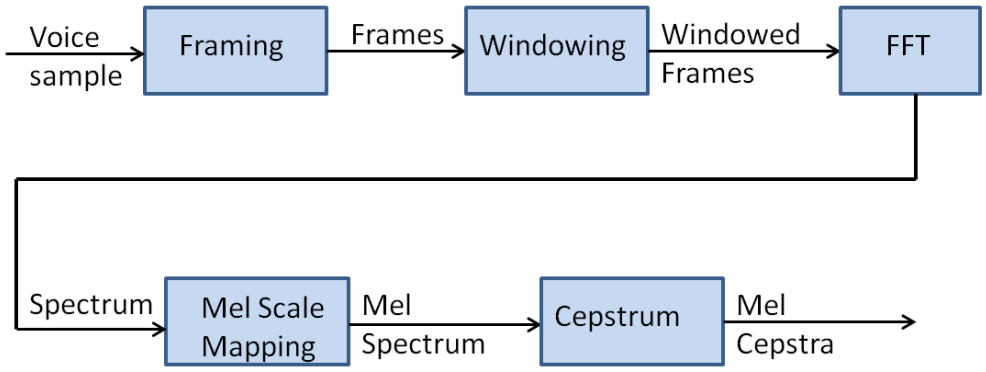


Figure 1: MFCC Block Diagram

The Mel spectrum of the signal is then separated into filter banks to represent the characteristic frequencies of the signal. The filter banks are evenly spaced on the Mel scale. The number of filter banks used will correspond to the number of MFCC coefficients produced. This parameter can vary, but it is typically chosen to be 20. [SUK12]

The log Mel spectrum is finally transformed back to the time domain using the discrete cosine transform. The coefficients that result from this transformation are the MFCC values that are used to represent the speaker's voiceprint. [DOM13]

2.2.2 CoMIRVA Library

Voiceprint Vault relies on the CoMIRVA library to extract features from audio samples. CoMIRVA is a Java library that contains implementations for many audio processing functions. This library includes a MFCC implementation that Voiceprint Vault uses to extract MFCC coefficients from audio samples that it receives. CoMIRVA is licensed under the GNU GPL and so as long as the Voiceprint Vault Server relies on this library, the source code will be freely available. [SCH11]

2.3 ADDITIONAL LIBRARIES

This system makes use of several libraries in order to accomplish larger objectives. Each library that Voiceprint Vault includes is discussed in the sections below.

2.3.1 Java Cryptography Library

The Voiceprint Server uses cryptographic operations to protect specific pieces of data that are crucial to user and server security. The user's password is hashed using SHA1 hashing and a per-user salt value, so that even if the App Engine server was compromised, the user's password would not be able to be determined from the password hash. This is described in greater detail in Chapter 4. All the functions required to implement this are included in the javax.crypto library. [JCP13]

2.3.2 Apache Commons IO File Upload Library

The voiceprint training recordings and the voiceprint verification recordings are both uploaded from the client application to the server as WAV files. The Apache Commons IO FileUpload library makes the task of receiving these files much easier. The files are included as part of a multi-part form by the client application. The FileUpload library assists in reading the file data from the HTTPS request as well as other metadata parameters that are uploaded along with the file.

2.3.3 WAV File IO Library

The Java WAV File IO library is used by both the client application and the Voiceprint server to write the WAV file and then read it on the server. In the client application, the Android audio recorder is used to record the user's voice. The raw audio data is then extracted and passed into the WAV library to create the actual WAV file. When the server receives the WAV file through the HTTPS request, the audio data is extracted from the WAV file and fed into the MFCC processing methods. The WAV File

IO library is free for use in private and commercial applications and is developed by Dr. Andrew Greensted. [GRE10]

Chapter 3: System Architecture

3.1 Overview of System Architecture

The Voiceprint Vault system uses a client-server architecture. This allows the authentication service to be integrated across a variety of platforms with the verification and user data stored in a central location. The server is hosted on Google App Engine and accepts authentication requests through secure HTTP. The following sections explore the technical aspects of this system design in more detail.

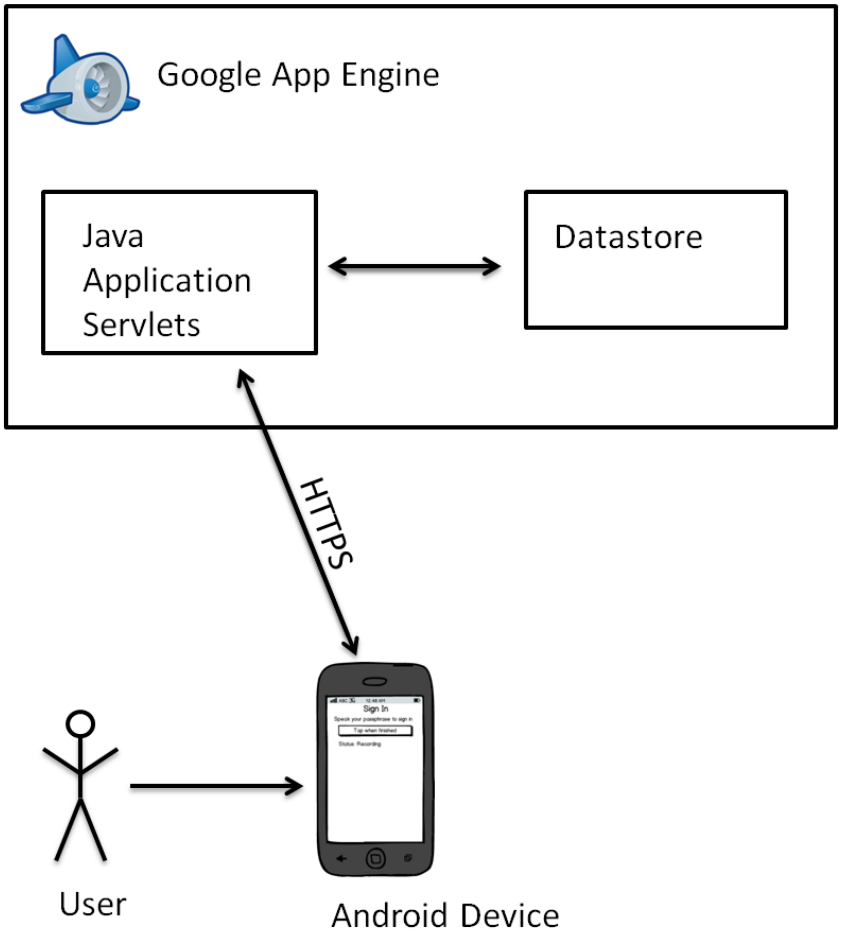


Figure 2: Client-Server System Architecture

3.2 APP ENGINE SERVER IMPLEMENTATION

The Voiceprint Vault server is hosted on Google App Engine to take advantage of the features discussed in Chapter 3. Several aspects of the server design are discussed in the following sections.

3.2.1 Datastore Hierarchy

The App Engine Datastore holds data in entities. Each entity is identified by its "kind" and contains named properties that store data or provide links to other entities. Every entity also has a key which identifies its parent entity if it has one. Using a hierarchy of entity kinds, the structure in Figure 3 was created to store all Voiceprint Vault data.

In the figure, the VoicePrintUsers entity kind holds the root entities, those without any parent. These entities have a property that identifies the repository name. This entity is used to isolate all the different client applications of the Voiceprint Vault system, so that multiple clients can use the same Datastore space without interfering with other clients' data. A VoicePrintUsers entity with property "depotName=VVSampleApp" would be the parent entity for all user data registered under that client. When a developer integrates Voiceprint Vault authentication into a new application, the first step is to set up a repository in the Datastore, which internally creates a new VoicePrintUsers entity with an associated repository name.

One User entity exists for each registered user of a Voiceprint Vault client application. This entity stores the user's email address, password hash, password salt, and security settings.

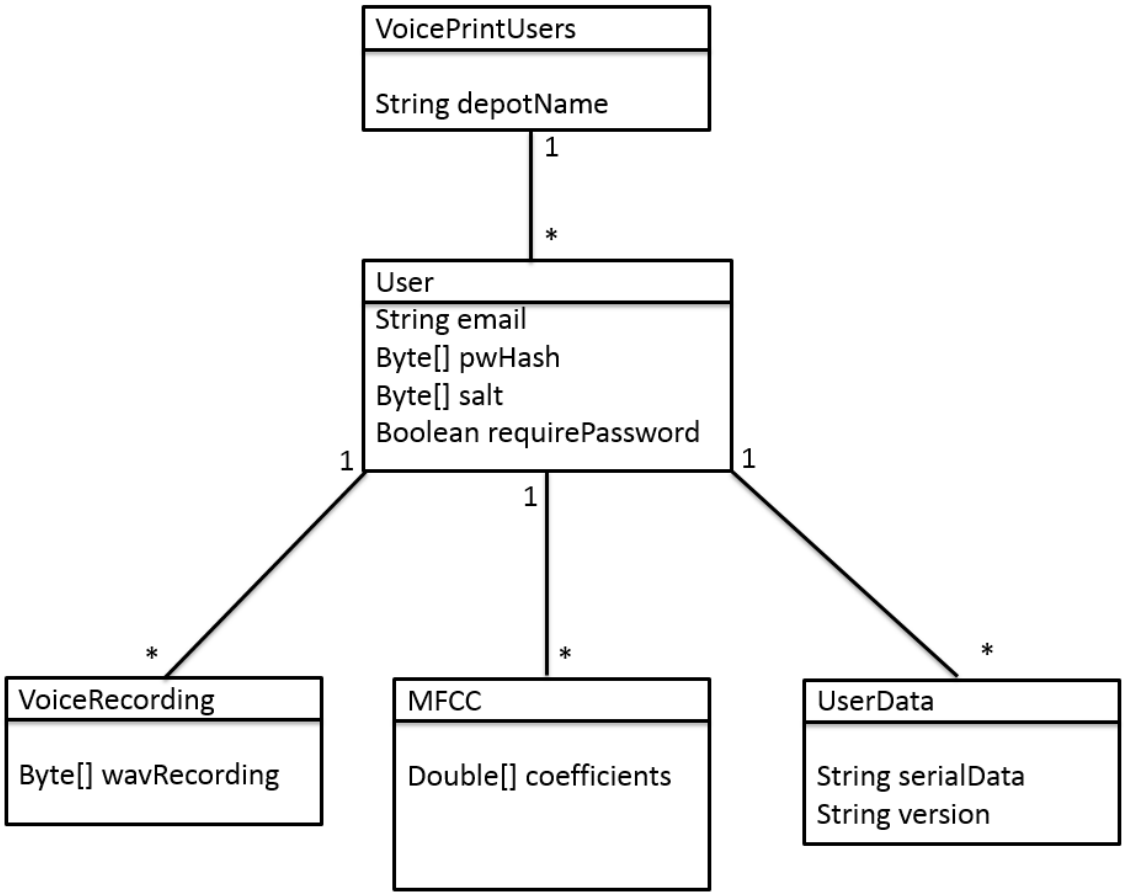


Figure 3: Hierarchical structure of Voiceprint Vault data in Datastore

The children of the User entity contain both the user data and their analyzed training recordings. The UserData entity holds serialized data from the client application that can easily be synched across devices, as well as a version to track updates to the data. The VoiceRecording entity holds a byte array that represents the WAV audio data from the user's training recording. The MFCC entity holds the MFCC coefficients computed from the user's training recording. The MFCC coefficients are computed prior to authentication attempts for faster response time from the server.

3.2.2 Servlets

Google App Engine uses Java Servlets to handle web requests. The Voiceprint Vault server defines a web.xml deployment descriptor that maps incoming URLs to the appropriate servlet to handle each request. The following servlets are used by the Voiceprint Vault server to handle common application requests:

- Receive New Account Servlet – This servlet is called when a new user chooses to create an account. It takes the email address, password, and security settings for that user and creates an account for them in the Datastore.
- Upload Recording Servlet – This servlet is called following a new user account creation when they upload their voice for their training recordings. MFCC coefficients are calculated for the new voice recording and then stored in the Datastore.
- Authenticate Voice Servlet – This servlet is called with a user's email address and voice recording in order to verify their voiceprint. The servlet computes the MFCC of the voice recording and then returns an access token to the user.
- Authenticate Password Servlet – This servlet is called if password authentication is required after the user has already authenticated their voice. After verifying the user's password against the password hash in the Datastore, an access token is returned to the user granting them full account access.
- Upload Database Servlet – This servlet is used to update user application data in the Datastore. In the example Android Application seen in Chapter 5, this servlet is called when the Save button is clicked from the home page, causing all updated user data to be stored on the server.

- Upload Settings Servlet – This servlet is used to update user account settings. The only global setting at this time is an option to require password or not. In order to update the account settings, a user must enter their password in addition to already being authenticated by their voiceprint.
- Client Registration Servlet – This servlet allows new application developers that wish to integrate Voiceprint authentication to register the application ID with the Voiceprint server.

3.2.3 Server Configuration

One of the major goals of this framework is to allow for many client applications to use this single voiceprint authentication server and customize it to the needs of their application. The server defines a strict protocol for communication and data storage, but the system design is intentionally minimalistic at this time. This will allow for future extensions, customizations, and allows for clients to already take advantage of some flexibility within their own client application. In this section, two desirable customizations to the server behavior are considered. Customizations to the client application are considered in Section 3.3.

3.2.3.1 Data Storage Preferences

The server currently provides a storage mechanism for user data, but it is far from ideal. The client application is able to upload data to the UserData entity, which is a child of the User entity in the Datastore. The UserData entity kind has two string properties: serialData and version. This is meant to allow client applications to serialize their data into a string and save it in the Datastore so that it can be synched and deserialized across

all user devices. The version field is intended to be used to differentiate between different versions of the serialized data across synched devices.

This mechanism has some clear weaknesses. A string is neither the most efficient, nor flexible means of storing data. This mechanism should be generalized to take a Java Object and allow the client application to handle merging of conflicting versions, since this will allow the behavior to evolve without server changes.

3.2.3.2 Security Settings

There are few security settings at this time, but they may be extended later. The only existing setting is whether a password should be required at sign-in time, and this setting is controlled by the user. This setting could very well remain in user control, but a client of the Voiceprint Framework may wish to require that a password always be used, and may also wish to customize other security parameters, such as the hashing and encryption algorithms used. These settings would ideally be made available to the client to control from a set of pre-defined values.

3.3 ANDROID LIBRARY IMPLEMENTATION

A Java library for Android is provided to assist with Voiceprint Vault integration in Android applications. It contains classes and methods required by all client applications that plug into the Voiceprint Vault framework. The library is packaged as a 22 kilobyte JAR file and is available for download by client application developers at https://voice-print.appspot.com.

Clients may also want to see a standard implementation of the typical account creation and login pages that are needed to give users full access to the Voiceprint Vault system. For this reason, a set of example activities and Android resource layouts are also available for download along with the Java Library. This example code is not strictly part

of the library, but is tightly integrated with the library and can be used by a client application with minimal changes.

3.3.1 Library Classes

The Voiceprint Vault Java Library contains several classes that are essential to the operation of any application using the Voiceprint Vault framework, and do not require modification. For this reason, they can conveniently be bundled into a JAR file for client application developers to download and integrate into their application without needing to alter the existing code.

The `UserAccountData` class encapsulates the concept of a user for the purposes of this framework and holds all the data members that need to be set in order to create a user in the Voiceprint Vault Datastore.

The `SavedUserManager` class is not essential to the operation of an application, but allows for convenient caching of users who have accessed the Voiceprint Vault from a given device. This class is used to cache usernames in a Sqlite database (provided that permission is given), for the convenience of the user on future logins. In the example application and as demonstrated in the example activities, the saved users can be chosen from a drop-down menu on the sign-in page.

The `VoiceprintUploadTask` class is an abstract class that is provided to assist the client app developer in creating http requests to send to the Voiceprint Server. This class can be used both for sending a simple set of parameters to the server, as well as uploading an audio file to the server for analysis.

The class is abstract, requiring that the developer implement one method in their own concrete subclass. The developer must control what code should be run after the upload is complete by implementing the `onPostExecute()` method. After a concrete

subclass of the `VoiceprintUploadTask` has been defined, this class can easily be used to upload a web request. The client code must instantiate a new object of the subclass and set the URL to which this request should be directed. The client code can additionally add a file and parameters to include in the request with the `addParameter()` and `addFile()` methods. At this point, the upload task can be executed and the client's handling code for the response will be run after the request finishes.

The `VoiceprintRecordTask` class is an abstract class that is provided to assist the client app developer in recording the user's voice for the purpose of training and authentication. Like the `VoiceprintUploadTask`, this class is abstract and must be subclassed before it is used.

The subclass must implement the `onPostExecute()` method which controls what client methods should be run after the recording is complete. To start a recording, the client code must set a file path for the recording and then execute the `VoiceprintRecordTask`. The recording can be stopped at any time by asserting the `stopped` member field of the class.

3.3.2 Example Activities and Layouts

In order to facilitate the use of this framework, example activities and layouts for an Android application are provided. An example application has also been developed to fully demonstrate an application using this framework, and can be seen in Chapter 5. The landing page, account creation pages, and sign-in pages used in this example application are provided as example code for client application developers.

These files are provided separate from the previously described Voiceprint Vault Java Library, because they may be modified by the user, whereas the classes in the Java

library are not intended for modification. The intent of this example code is to provide developers with as much or as little customization to the default pages as they desire. Both the activity classes and the resource layouts are included in this example code download, so a developer could include these files in their application and have a fully functional application with very little work.

The only class in the example code that must be updated by the app developer is the `VVClientProperties` class. This class simply holds some global properties used by the other activities. One such property is the voiceprint_client_depot, which is used to reference an application-specific identifier that is used in the Voiceprint server to separate all relevant user data and settings from other applications using this framework.

The remaining activities and layouts can be skinned to conform to the look-and-feel of an application, or can even be reorganized such that authentication and data updates occur in different portions of the application as needed. Chapter 6 describes more specific instructions for integrating this example code into a client application.

3.4 CLIENT-SERVER COMMUNICATION

The client application must communicate many user actions to the App Engine server to be processed because all security is implemented on the server side. The figure below depicts the most prominent exchanges that occur between server and client for a user to set up a new account and then access that account with voice authentication.

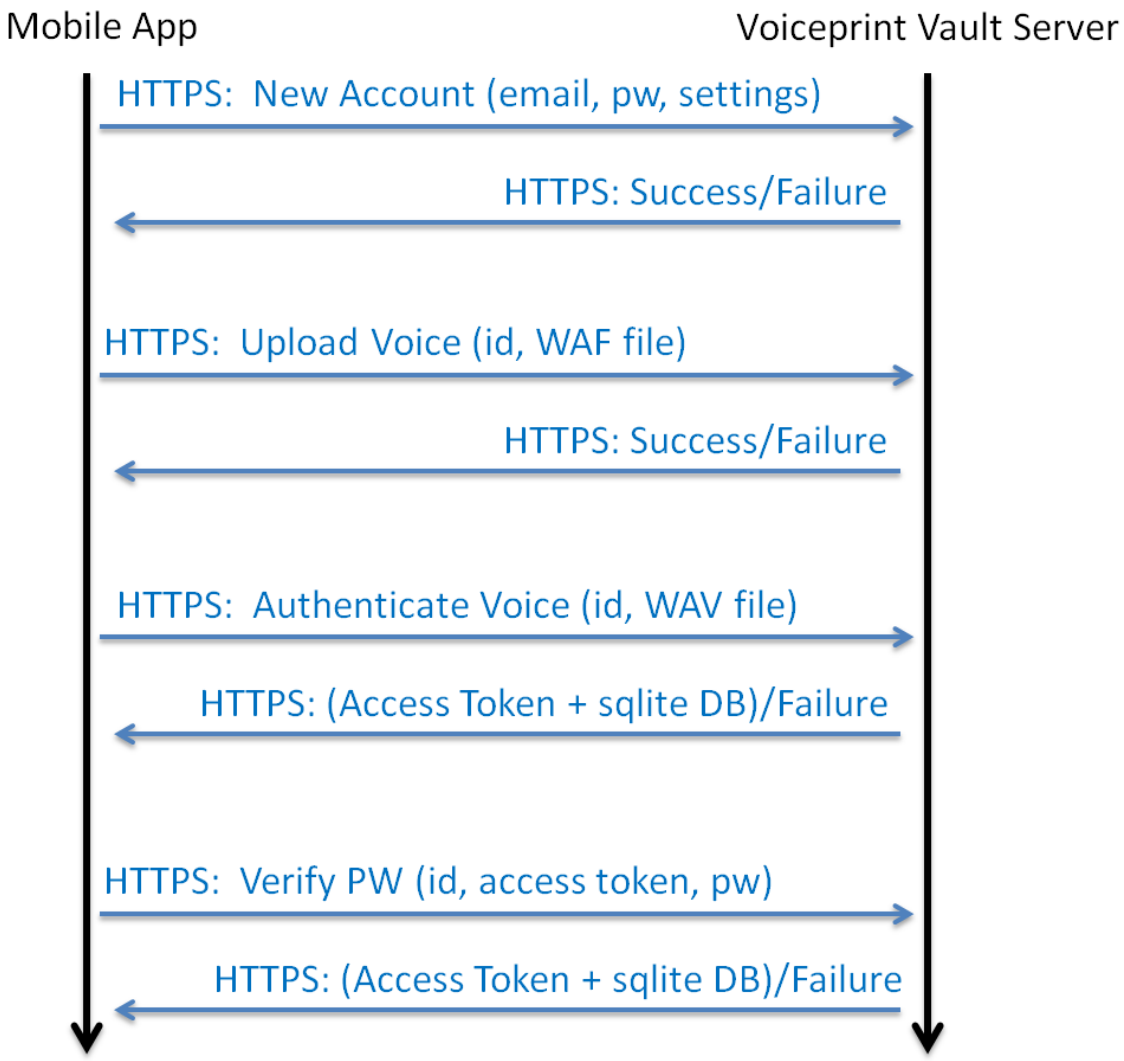


Figure 4: Messages between client application and Voiceprint Vault server

3.4.1 New Account Message

Users must initiate their usage of Voiceprint Vault by creating an account. After they have entered their email, password, and authentication settings into an Android form, the application sends this data through a secure HTTP transmission to the App Engine server. The server verifies the submitted data and creates an account for that user in the Datastore. A failure message is returned to the client if the account is rejected for

any reason (duplicate/invalid email, corrupted data, missing fields). Otherwise, a confirmation that the account was created is returned to the client (also through HTTPS).

3.4.2 Upload Voice Message

After creating an account, users must train the system with their voice. Users are prompted to record a passphrase on their mobile device. Their recording is saved as a WAV file and then transferred to the server, along with their email address, which serves as their identifier. The App Engine server analyzes the incoming audio sample, computes the MFCC coefficients for future verification, and stores these values in the Datastore. Upon successful computation, the server returns a confirmation to the user and their account creation is complete.

3.4.3 Authenticate Voice Message

After creating an account and uploading training recordings, users are able to log in to their account and access the application. In order to access their account, users must authenticate with a spoken phrase. After recording their phrase, the client app sends the voice recording and email address to the App Engine server. The voice recording is analyzed for corresponding MFCC coefficients. If the coefficients are found to be within a given threshold from the training recording coefficients, then the user is given an access token generated by the server that will grant them access to their existing account.

3.4.4 Verify Password Message

In the event that a user is required to supply their voiceprint and password to gain access to the account, then the access token returned to them in the previous step is incomplete. The user must also supply their password in order to gain full access to the account. After the app communicates the password to the server, the server takes a digest of the password and verifies that it exactly matches the saved password digest. As with

all communications between server and client, all of these messages are transferred over HTTPS so that no intermediate node in the network routing can decipher anything from these messages.

3.5 AUTHENTICATION TIMING

The round trip time was measured for a Voiceprint authentication request. In order to avoid user frustration, the time delay for an authentication request must not be prohibitively slow. The elapsed time began just before the client application sent a recording to the server and ended after the authentication response was received from the server. In that time, the Voiceprint Vault server on App Engine receives the request, extracts the recording, performs the MFCC calculations for the voice recording, compares the result to the user's training recording, and returns the result to the user. The user will either receive an access code back on successful verification or an error message indicating that login failed. This measurement is important from a usability perspective because it measures how long the user has to wait with their device in their hand while the server is processing their request.

This test was performed with a HTC One X phone running the reference implementation of a Voiceprint Vault client app. The round trip time was measured over five trials on a 4G LTE network, and the average elapsed time was 585.4 milliseconds.

Over the same five trials, the computation time in App Engine was measured. The average time required for the authentication request to be handled was 118.6 milliseconds. The cost of computation time in App Engine is $0.08 per hour, so each authentication request that takes 118.6 milliseconds costs 0.26 millicents.

3.6 SOFTWARE METRICS

Software metrics were computed for the Voiceprint Vault server and the Voiceprint Vault Java Library. These metrics are displayed in the following tables.

Metric	Value
Total Lines of Code	3526
Number of Packages	3
Number of Classes	22
Number of Methods	180
Number of Static Methods	38
Number of Static Attributes	41
Number of Attributes	38
Average Number of Parameters	1.606

Table 1: Software metrics for Voiceprint Vault server

Metric	Value
Total Lines of Code	2933
Number of Packages	4
Number of Classes	54
Number of Methods	210
Number of Static Methods	6
Number of Static Attributes	266
Number of Attributes	100
Average Number of Parameters	1.236

Table 2: Software metrics for Voiceprint Vault reference implementation

Chapter 4: Security Features

4.1 VOICEPRINT VAULT SECURITY FEATURES

Voiceprint Vault is intended to provide secure access to private data across a variety of applications, so it is important that the security model does not have any exploitable vulnerabilities. All of the security must be implemented on the server side, so that the system is protected from fraudulent applications or modifications to a legitimate client application. The server code is protected by Google account security, so that only authorized developers can deploy to the App Engine server. The following sections discuss some elements of the security model for this system.

4.1.1 Login Requirements

All users of this application are required to upload a voice sample at account creation time and authenticate with their voiceprint at login time. When a user attempts to access their account, they must enter their email address and are then prompted for their voiceprint sample. Their recording is uploaded to the server and compared against their training recording. If the voice passes this verification phase, then the user is granted account access.

Users are given the option at account creation time to require a password also for access to their account. This makes the authentication scheme a true multifactor authentication. If the user is verified at the voiceprint authentication stage, but also requires a password for access to their account, then they are directed to a password prompt by the client application. It is only after a correct password is entered and verified by the server that the user has full access to their account.

4.1.2 Password Storage

In any web application, it is essential that the user's password is not ever stored as plaintext on the server. This ensures that if anyone ever gains access to the server, the users' plaintext passwords are not revealed. Instead, a hash of the password is computed upon account creation and stored in the Datastore. When the user logs in to their account, their password is sent to the server and then hashed again. The server verifies that the password hash matches the user's stored password hash.

It is further important that the password hash is combined with a random salt value that is created at account creation time on a per user basis. The salt that is combined in the hashing algorithm serves to protect against attacks using pre-computed rainbow tables and also slows down brute-force attempts to crack the password.

Voiceprint Vault uses both of these features. On account creation, a per-user 1024 bit salt value is randomly generated. That salt is stored with the user's account in the Datastore. The password given by the user is then run through 2000 iterations of the PBKDF2 algorithm using a SHA1 hash and the 1024 bit salt value. This means that the password goes through many intermediates steps before a 1024 bit key is eventually produced. This key is stored with the user's account also and used to authenticate users at login time.

4.1.3 Access Tokens

When a Voiceprint Vault client authenticates with the server, they have an open session that allows that user to view, modify, and delete their private data. Access tokens are used by the server to store the user's privileges and be included with future requests on the user's behalf. Access tokens are an elegant way of securely controlling data access without too much bookkeeping on the server side.

Access tokens rely on encryption with a private key only known to the server. When a user successfully authenticates with their voiceprint, a string is created that includes a user ID, the user's privilege level, and an expiration date for this access token. That value is then encrypted by the server using the Advanced Encryption Standard (AES) with a secret key.

The encrypted access token is returned to the user as a reply to their authentication request. If the user has only authenticated with their voiceprint, and their account requires password authentication also, then the access token's privilege level is used to indicate that they have not fully authenticated yet. After entering a correct password, a new access token will be returned with an updated privilege setting.

The client application includes the access token in all requests that are sent to the server after their initial login. The client itself is unable to decode the access token, but before processing any request from the client, the server will verify that the included access token is valid.

All of these features work together to provide a secure framework that is safe for account holders and application developers and preserves user privacy even from the application developer. Some possible extensions to the security model for this application are considered in the evaluation of this system in Chapter 7.

4.1.4 Network Attacks

The App Engine Framework provides a number of security benefits, including resiliency against a variety of common network attacks. App Engine servers are replicated locally and across geographically dispersed regions. This makes the entire system much less vulnerable to a denial of service attack. If one App Engine server is

attacked, the web requests can be routed to another server until the attacking IP address has been blocked.

The underlying Secure HTTP protocol protects against man-in-the-middle attacks. All messages between the Voiceprint Vault server and client are transferred using HTTPS. This means that a secure key exchange occurs before any data is transferred that cannot be intercepted by an intermediate network node. Following the key exchange, all network traffic is encrypted with a secret key, so an intermediate node cannot intercept any messages or pose as the client or server.

4.2 TARGET AUDIENCE

Security of personal data is clearly an important issue in this application. The Voiceprint Vault system aims to provide both security and convenience to users, requiring a convenient biometric voice sample from all users, and an optional password. When creating an account, users should consider what data they are storing in the application and how critical that information is.

For the highest security, the optional password should be required by users when they create their account. This completes the multifactor authentication scheme for this system by requiring a knowledge factor and an inherence factor. Users that expect to be targets of data theft, that have critical data stored in their account, or whose voice could easily be recorded are recommended to require a password on their account at all times.

For users that are not storing critical data and do not expect to be the targets of data theft, they have the option of not requiring a password at login. This is meant to be the ultimate in convenience and security, since a user can access their private data in front of others without having to cover their device as they enter a password. While a voice

could easily be recorded by an acquaintance, some users may not expect anyone they know to target them and are comfortable with voiceprint authentication only.

Chapter 5: Reference Implementation of Client Application

5.1 Overview of Application

In order to demonstrate the functionality of this authentication service and the ease with which a client can integrate Voiceprint Vault authentication into their application, a reference implementation was created that uses the Voiceprint Vault Java library. This example application was created for Android because Android OS holds over 50% of the smartphone OS market share. [COM13] Additionally, this promoted code reuse between App Engine and Android since applications for both can be developed in Java.

The example application allows users to protect access to private data with voice authentication. After authenticating their voiceprint, the user can create and edit notes, tasks, and reminders through this application. All user data is stored in the Datastore when the user exits the application and data is automatically synched across all devices that the user runs this application on. This application is an example of the kind of personal data that is convenient to keep on a mobile device but should be protected in the event of a lost or stolen phone.

5.2 User Interface

The application was designed with a simple and intuitive user interface. Screenshots from the different pages of the application are included below with descriptive comments.

5.2.1 Landing page and Create New Account

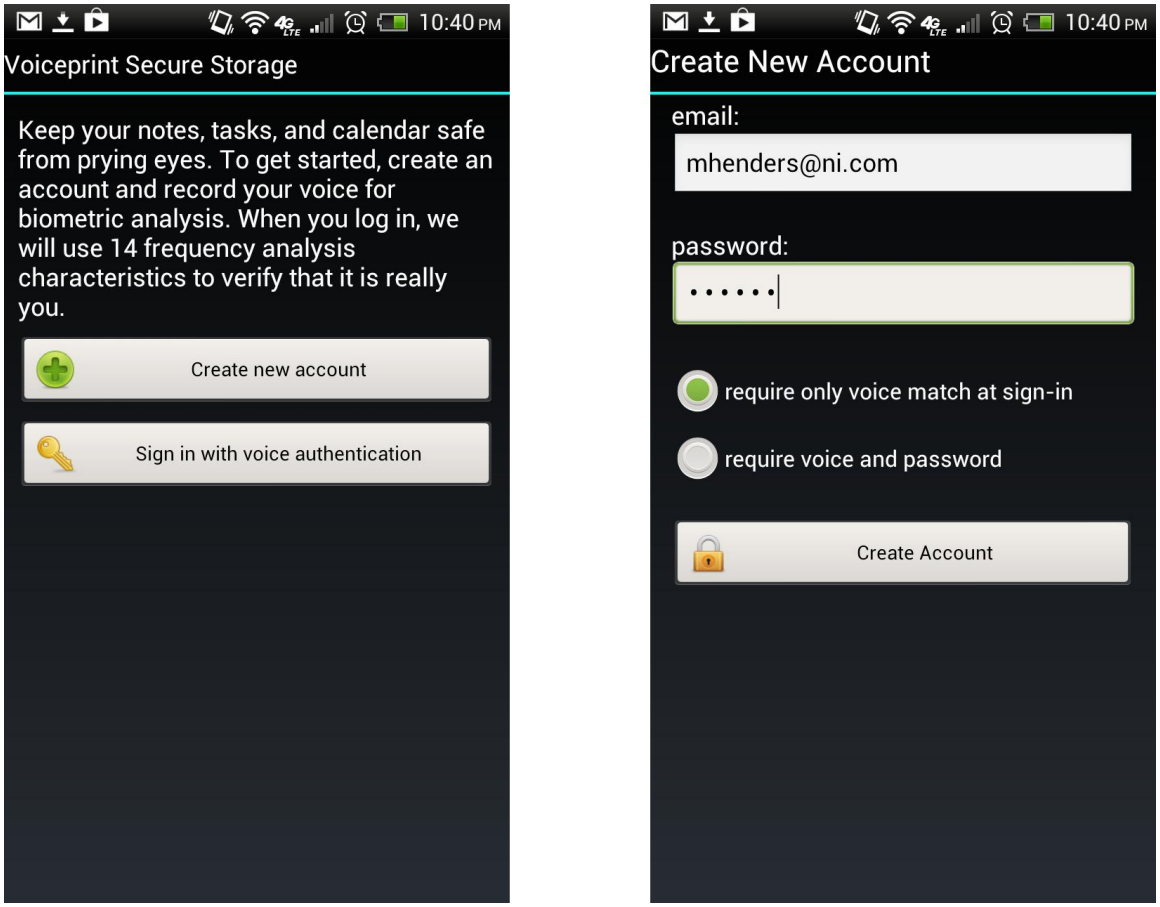


Figure 5: Application landing page and Create New Account page

When the user launches the application, the landing page on the left of the figure above is what they see. The landing page provides options to create a new account or sign in to an existing account. If the user chooses to create a new account, it leads them to the page on the right of the figure above. The user is prompted for an email address, password, and whether they want their password to be required in addition to their voice at sign-in time.

5.2.2 Sign In Pages

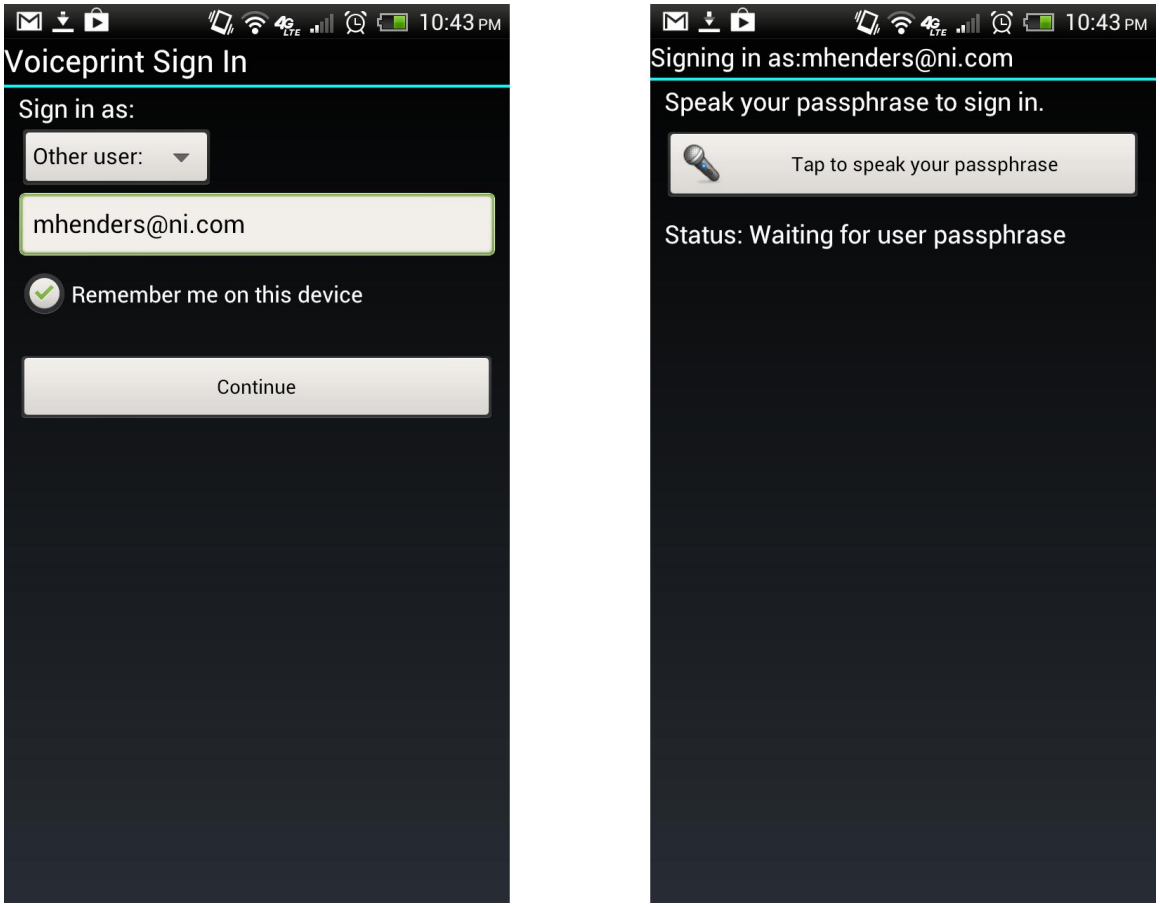


Figure 6: Voiceprint Sign In page and Passphrase Verification page

After the user has created an account, they may sign in to their account by selecting that option from the landing page. The Voiceprint Sign In page asks for an email address linked to their account and allows the user to selectively save their address for future account access.

Once the user has entered their email address and continued, they are prompted for their passphrase. The user must tap the button to begin recording, speak their passphrase into the microphone, and then tap the button again to stop recording. After the passphrase has been recorded using Android recording libraries, the WAV file is

uploaded to the Voiceprint Vault Server for verification. The server responds with a success or failure message and an access token in the event of success.

5.2.3 Password Page and Voiceprint Home Page

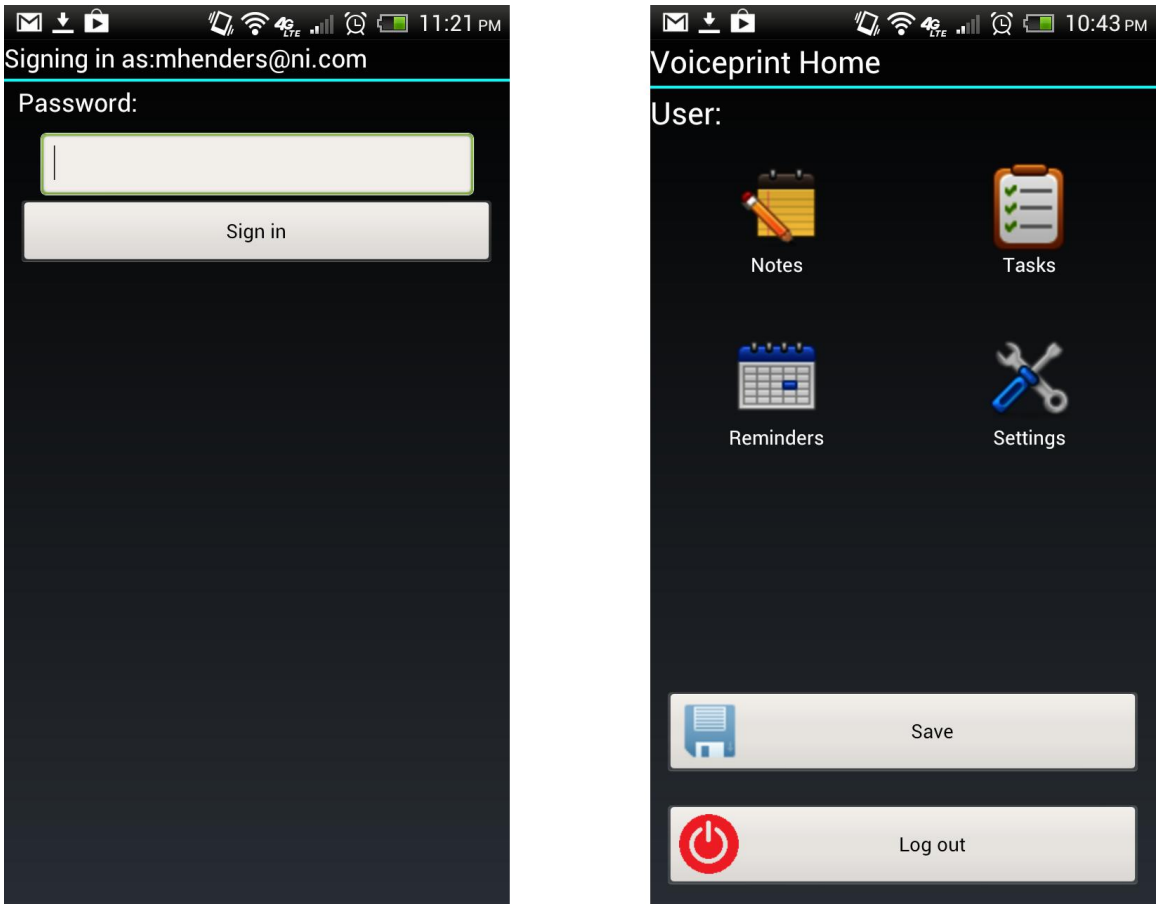
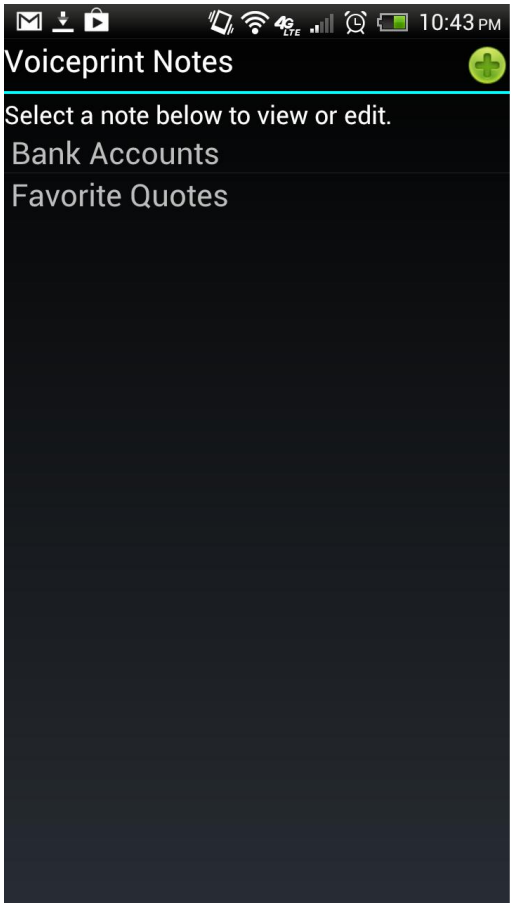


Figure 7: Password Verification page and Voiceprint Home page

If the user's voice is verified, but they also require a password to access their account, then they will be redirected to the Password Page. After entering a password and clicking the Sign In button, another request will be sent to the server. After verifying the password, a response from the server will indicate success or failure and provide an access token in the event of success.

After the user has been verified and received an access token, they are directed to the Voiceprint Home page (on the right in the figure above). The home page gives the user access to all their private data (notes, tasks, reminders, and settings). The user can also save their data, which uploads their latest revisions to the App Engine Datastore. If the user logs out, then their latest changes will be uploaded to App Engine and then their access token will be destroyed, requiring them to log in again for further access to the application.

5.2.4 Notes Home Page and New Note Page



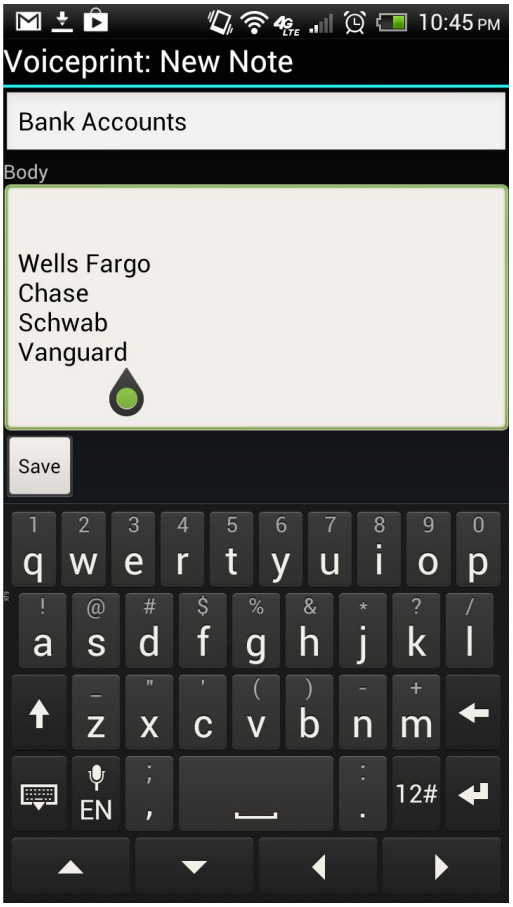


Figure 8: Voiceprint Notes page and Voiceprint Note Creation page

The Notes Home Page on the left shows the user all the notes that they have saved in their account and the New Note page shows the dialog for creating a new note.

5.2.5 Reminders Home Page and Edit Reminder Page

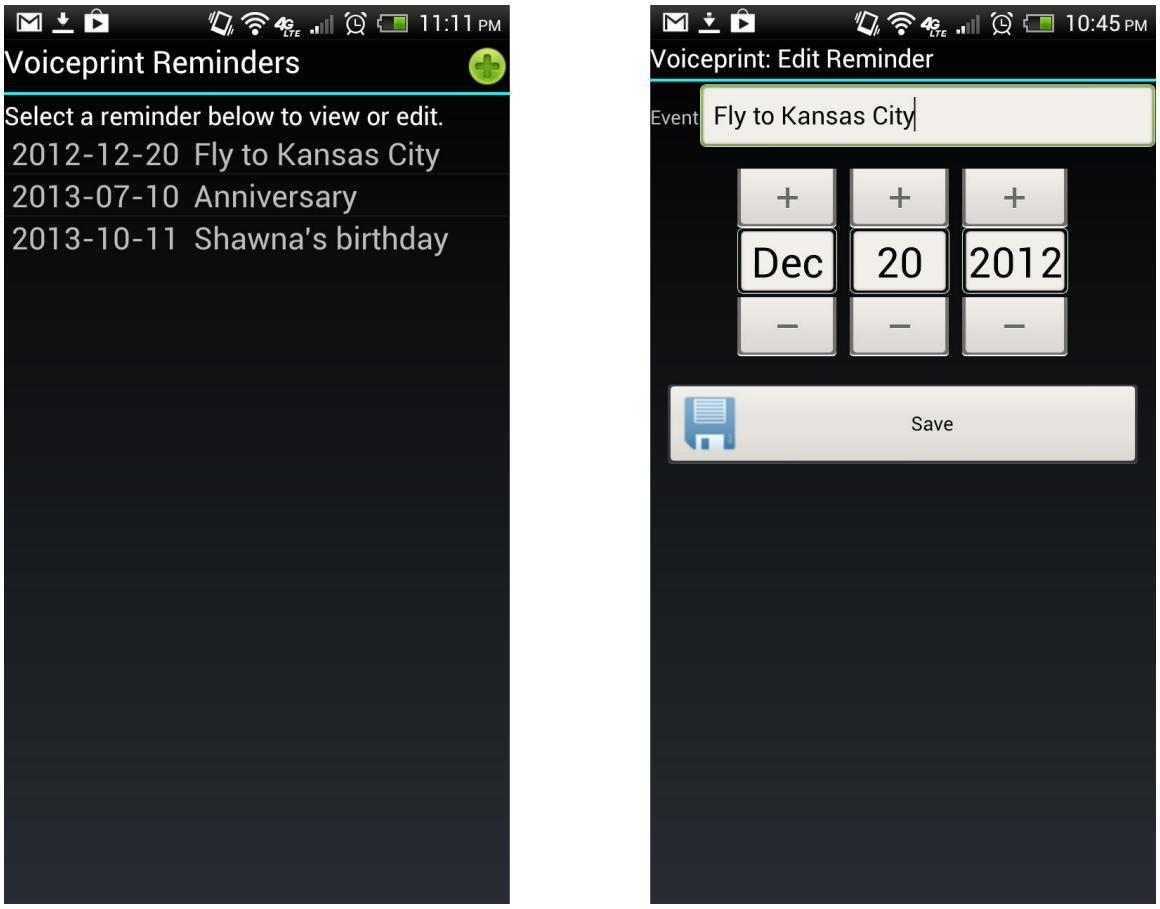
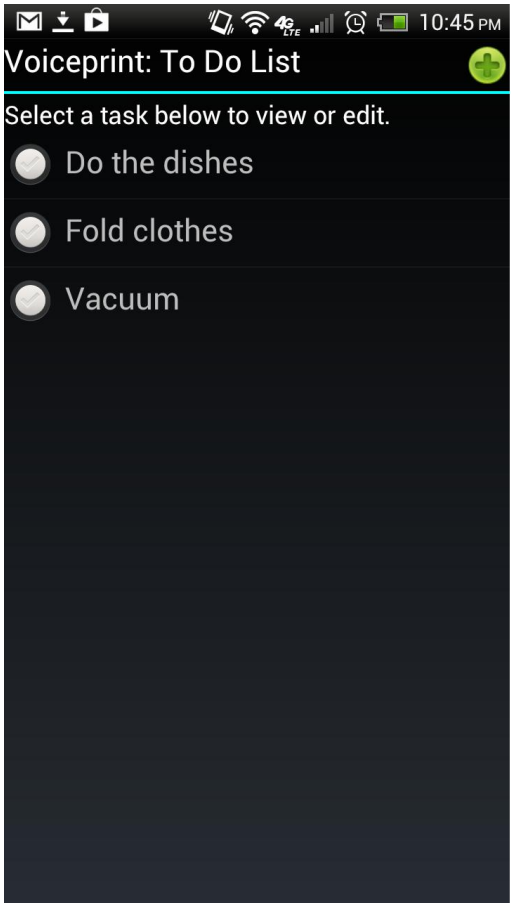


Figure 9: Voiceprint Reminders page and Voiceprint Edit Reminder page

The Reminders Home Page on the left displays all the reminders the user has created and the Edit Reminder Page shows the interface for editing an existing reminder.

5.2.6 Tasks Home Page and Settings Page



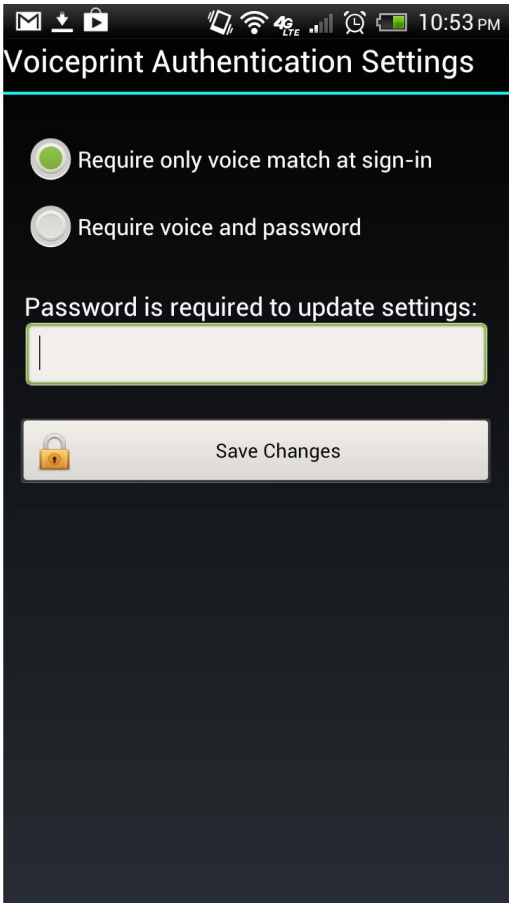


Figure 10: Voiceprint Tasks page and Settings page

The Tasks Home Page displays all the tasks that the user has awaiting completion. The Settings page allows the user to change their current security settings and requires that the user enter their password to authorize any changes.

5.3 USE OF VOICEPRINT VAULT AUTHENTICATION

This application integrates the Voiceprint Vault authentication system just like any client application would. This application uses the default Landing Page, New Account pages, and Sign In pages that are provided in the example layouts included with the library. The activities that control these pages are provided as examples also, and

were customized to open the Home Page following authentication. The Settings page used in this application is also provided as an example layout for any client applications that would like to allow users to modify their account settings. The activity for the settings page was also modified to fit into this application's navigation scheme.

Much of the included example code can be included in any existing Android application that integrates Voiceprint Vault authentication. There are a few components that need to be updated by the client application developer in order to complete their integration with the Voiceprint Vault framework. These customizations are described in detail in Chapter 6.

Chapter 6: Integration Instructions for Client Applications

6.1 Overview of Integration Requirements

A major goal of this project is to ease the integration process for client application developers. In order for this system to attract clients, the steps required for developers need to be well documented and easy to follow. This section details all steps required by application developers to plug their existing application into the Voiceprint Vault authentication framework. The instructions are specific to Android applications but can be generalized to other platforms as well.

6.1.1 Registering the client application with the Voiceprint server

The Voiceprint Vault server is designed to allow many unique client applications to use the same Datastore while keeping user data protected from other client applications. To this end, the Datastore is constructed with a hierarchical ordering, where all voiceprint records, user data, and privacy settings for a given client application are stored underneath a dedicated parent entity for that client. For the example application detailed in Chapter 5, this means that all relevant data is stored under the "VV_Example_Application" key.

The first step in integrating Voiceprint Vault authentication into an existing application is to register an application ID with the Voiceprint Vault server. This can be done from the client registration page, at https://voice-print.appspot.com/client_registration.

6.1.2 Including the Voiceprint Vault Java Library

All of the classes and methods required for an application to communicate with the Voiceprint Vault server are made available as a Java library. The Voiceprint Vault Java library is available as a JAR file https://voice-print.appspot.com. To include this library in a client application of the Voiceprint Vault framework, the JAR file should be downloaded and referenced in the build path of the Android application. All of the classes and methods included in this library are not intended to be modified and therefore the source code for these classes is not exported in the library.

6.1.3 Using the Example Activities and Layouts

The landing page, account creation pages, and sign-in pages from the Voiceprint Vault example application (as seen in Chapter 5), can all be reused in client applications with little modification. It is expected that most clients that integrate the Voiceprint Vault framework will display a similar landing page giving users the option of creating a new account or signing in to an existing account. Regardless of which option the user chooses, it is also expected that the sequence of pages encountered after the landing page would be similar for most client applications as it is for the example application. For that reason, the source code for the activities and layouts that control these pages is provided for free use and modification in all client apps.

The source code for these activities can be downloaded from https://voice-print.appspot.com along with the Voiceprint Vault Java library. The layouts should be included under the layouts resource directory for the client application and the activities should be included with the rest of the application's Java source code. The activities and layouts can be modified in any way that is desired by the client developer. It is expected that the layouts are likely to be modified to fit the application's look-and-feel. If no changes are desired, then the activities can be used as they are.

6.1.4 Referencing the Client Application Key

The VVClientProperties class is the only class that must be modified. This class references the client application identifier that was used when registering with the Voiceprint Vault Server as described in Section 6.1.1. The same key that was used when registering with the server should be used in the VVClientProperties class so that all web requests from this client application reference the correct account on the Voiceprint Vault server.

6.1.5 Including Class References in Android Manifest

The AndroidManifest.xml file includes a list of all activities that are part of a given application. As such, this list needs to be updated with the name of all the example activities that are being used in the application. This file also references a main activity that is the first to be launched when the application starts. If the example landing page that gives users the option to create a new account or sign in is going to be used as the startup activity, then it needs to be marked as the main activity in the manifest file.

6.1.6 Launching Custom Activities from the Example Activities

The sign-in activities provided in the example code launch the home page from the example Voiceprint application after a user has been successfully authenticated. Since this home page will not be part of a client application, the reference to this activity should be updated to refer to the appropriate page in the client application.

Chapter 7: Evaluation of Voiceprint Vault Framework

7.1 PARAMETER OPTIMIZATION

The behavior of the MFCC algorithm that extracts characters from audio samples can be controlled with various parameters. The effect of modifying these parameters was researched with the aim of finding the optimal parameters for speaker verification. The objective of speaker verification is to recognize legitimate users and reject fraudulent users, so this study sought to find the point at which the difference between legitimate user verification and fraudulent user verification was the greatest. Due to the natural fluctuations in the human voice, there is always some chance of a legitimate user being rejected, but the goal of this study was to minimize the probability of this happening.

These tests were conducted using a sample set of users and voice recordings. The sample set included a total of 77 recordings from 14 unique users. The users were chosen to roughly approximate the range of voices you would find in a larger population. There were seven men and seven women in the set of users. Some users were related to each other while others were not. The age of the users ranged from 24 to 86.

7.1.1 Number of MFCC Coefficients

The MFCC algorithm applies a non-linear mapping of the frequencies in an audio sample to extract unique characteristics of a user's voice. The characteristics are concisely represented in a set of coefficients that are produced by this algorithm. Traditionally, MFCC algorithms use 20 coefficients. In this project, the effect of varying the number of coefficients was studied. A speaker verification test suite was run over the entire Voiceprint Vault database while changing the number of MFCC coefficients used in each successive test

In order to find the optimum verification results with each number of coefficients, the probability of an authorized user was compared against the probability of an unauthorized user. A difference in probabilities was taken (between authorized user and unauthorized user), and this difference was plotted. This test was run using everywhere from five to 22 coefficients for verification, and it was discovered that using 14 coefficients yielded the best results.

The results of this evaluation can be seen in the table below. The table lists the number of coefficients used, the maximum difference found between authorized user acceptance and unauthorized user acceptance, and then includes the threshold that yielded that maximum value. Using 14 MFCC coefficients yields the optimum difference for authorized users. These results were obtained when using an acceptance threshold of 0.30 for speaker verification. This indicates that using 14 MFCC coefficients with a 0.30 threshold gives legitimate users the greatest chance of being verified, while minimizing the probability that a fraudulent user will be verified.

Number of coefficients	Max difference for authorized vs. unauthorized users	Threshold used for best results
5	0.68	0.20
6	0.67	0.20
7	0.72	0.20
8	0.74	0.25
9	0.79	0.25
10	0.72	0.30
11	0.75	0.30
12	0.79	0.30
13	0.81	0.30
14	0.82	0.30
15	0.78	0.40
16	0.79	0.35
17	0.80	0.35
18	0.77	0.30
19	0.75	0.40
20	0.78	0.40
21	0.80	0.40
22	0.76	0.50

Table 3: Verification results with varying numbers of MFCC coefficients

7.2 PASSPHRASE OPTIMIZATION

Another variable that affects the performance of a speaker verification system is the passphrase spoken by the user during the training and verification phases. Initial users of the Voiceprint Vault system were asked to record several different passphrases for testing purposes. These recordings ranged from a simple "Hello, my name is…" to longer phrases that aim to capture more of the unique characteristics of a user voice and include most of the phonemes in the English language.

To test the effect that different passphrases had on the speaker verification, a test suite was run over the full set of recordings. In each test that was performed, a subset of the recordings in the database was selected. All combinations of recordings from that set

were then compared against each other, eventually yielding a single value that represented the difference between two recordings. In each case, the results were logged and compiled. The results were plotted against threshold values, where the threshold value determines what the maximum difference can be between two recordings and still have them identified as being from the same user.

The graph below depicts the results of the passphrase optimization tests. The graph shows at incremental threshold levels, what is the probability based on the sample recordings that a legitimate user will be verified and an impostor will be rejected. The red line shows the probability of an Invalid user being verified (referred to as a false positive). In the case where an impostor attempted to authenticate another user's account, the impostor would upload their voice and the server would compute a difference between the impostor's voice and the legitimate user's training recording. Depending on the threshold being used, the impostor will either be accepted or rejected. In the graph below, it is clear that at a threshold of 0.2 on this arbitrary scale, an invalid user has almost no chance of being verified, but if this threshold is raised up to 1.5, then the impostor has a nearly 100 percent chance of being incorrectly verified.

Similarly, the green line depicts the probability that an authorized user will be verified. For this test data, only recordings from the same user are considered. For each set of recordings, a difference between the recordings is computed. Since these recordings are all from authorized users, it is desirous that the difference computed is very small, but due to natural fluctuations in human vocal chords, their will likely be some difference detected. In the graph below, an authorized user has about a 15 percent chance of being authenticated when a threshold of 0.25 is used, but over a 90 percent chance of being verified when a threshold of 1.0 is used.

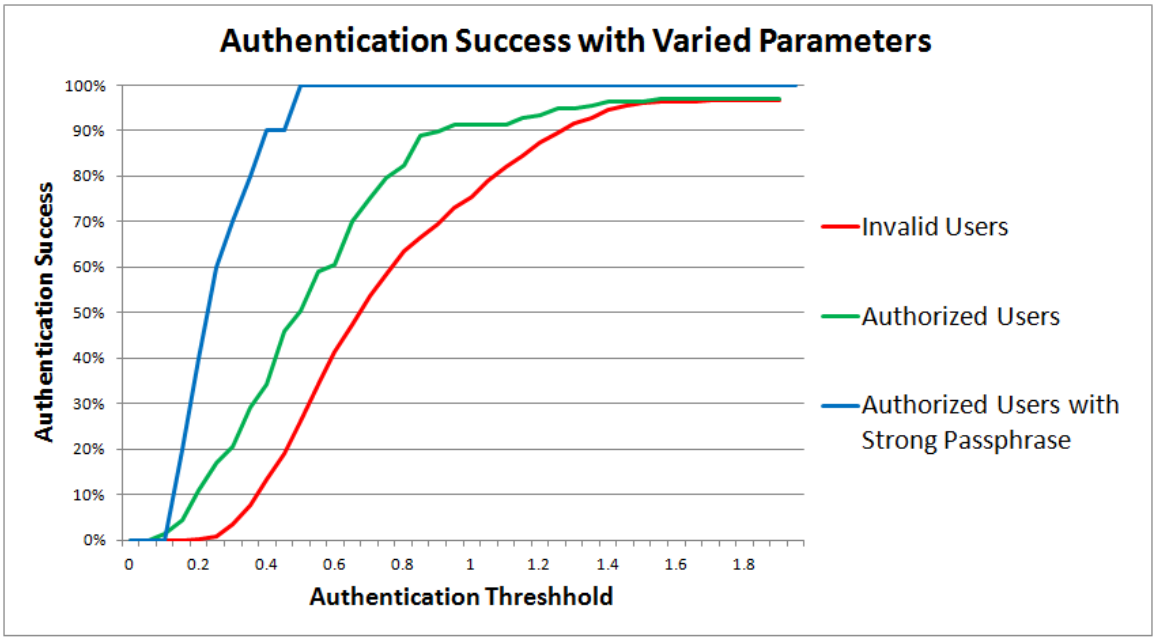


Figure 11: Speaker verification results with varied passphrase requirements

The purpose of this test is not just to compare authorized users and invalid users, but also compare specific passphrases spoken by legitimate users. The data for the green line was based on any two recordings from the same user being compared. However, if we assume that the legitimate user is going to speak the same passphrase every time they authenticate and that the passphrase is going to be about 10 syllables long, the authentication rate improves drastically. The blue line above depicts this scenario and shows that with a threshold of 0.4, an authorized user has nearly a 90% chance of verified while an invalid user has only about a 10% chance of being falsely authenticated. It is clear that setting some restrictions on the passphrase greatly improves the reliability of this authentication system. Since the probability of unauthorized users being verified climbs significantly above the 0.4 threshold, it is considered insecure to use any higher threshold value.

7.3 VIABILITY OF AUTHENTICATION SYSTEM

The passphrase optimization tests are a good indicator of the accuracy of this authentication system at verifying the correct ct users without accepting impostors. In the original results where the user was not encouraged to use a strong passphrase, the authentication system proved to be very weak. A user could simply say their first name as their passphrase, and this did not produce a long enough passphrase to identify sufficiently unique characteristics of their voice. For this reason, a legitimate user was only about 20% more likely to be verified than an impostor.

After encouraging a longer and consistently used passphrase from all users, the results were much better, even when the same passphrase was used by multiple users, because this authentication system is text-independent. These results showed that a legitimate user now had about a 90% chance of being authenticated, compared to an impostor having roughly a 10% chance of being authenticated. These results were much stronger, but it is still undesirable that an illegitimate user has a 10% chance of being verified against another user's account.

7.3.1 Options for Improved Verification

The resulting system with a 10% chance of allowing an impostor account access does not provide strong enough authentication to make it commercially viable for most organizations. This represents too much risk of intruders accessing private data. However, there are several steps that could be taken in the future to provide stronger authentication in this system and make this a commercially viable solution:

- Improve feature matching technique to provide more accurate verification
- Verify additional factors, such as GPS, device identifiers, user behavior patterns, etc.

- Impose longer passphrase requirements on the user and additional training recordings
- Require the user to repeat specific words and use text-based speaker identification
- Require the use of a password in addition to the speaker verification

All of these options have been considered as potential steps to improve the accuracy of Voiceprint Vault speaker verification. The first option is to improve feature matching techniques in the Voiceprint Vault server. As discussed in Section 2.2, this option could result in more accurate speaker verification, but was not implemented in this service in the interest of development time. This step would be the preferred method for improving verification accuracy because it does not interfere with any of the other goals of this authentication service and does not inconvenience the user.

Another option is to use additional factors to identify the user and confirm that they are who they claim to be. A few examples of additional factors that could be used are GPS coordinates, verifying the device they are using, and matching user behavior patterns with previous activity. All of these approaches could combine with the existing verification algorithm to improve the accuracy, but they are in contradiction to objectives of this authentication service. Additional factors could make the service more brittle and would be an annoyance to users when they do not perform as expected.

Added restrictions on the length and number of passphrases that a user must repeat are another viable solution to improve the accuracy of speaker verification. Without further tests, it is unknown how much the verification accuracy could be improved with this method, but lengthening the passphrase has proved effective at improving the accuracy already. This solution is not optimal, because it does represent additional and possibly cumbersome requirements on the user.

Similarly, the user could be required to repeat specific phrases that have been recorded in the past that are then analyzed by text-dependent verification algorithms. This approach would also place more requirements on the user, leading to a less ideal user experience. The additional requirements may be worth the cost to the user if they provide enough improved accuracy to the verification process.

The final approach is to use the existing password feature of the Voiceprint Vault framework to improve verification accuracy. It would be ideal if the user were not required to use a password to log in, but until the speaker verification results are improved, this approach is recommended. The combination of speaker verification and password is very secure, because even if an impostor can imitate the user's voice close enough to be authenticated, they must also know the user's password. This combination has all the security of existing password-based systems, plus the added security of voice verification. Despite the less than desirable accuracy, the voice verification adds great value to the authentication by posing a significant challenge to users who would attempt an automated brute-force password attack.

Client applications already have the option to require a password for authentication, and it is recommended that this option is checked for all but the most casual users of this application that are less concerned about the privacy of their account.

7.3.2 Applications for this Technology

This authentication system has a wide range of potential applications. Systems such as this are already being used in telephone banking and in some cases to log in to a mobile device. However, there is a vast array of other applications that could benefit from this technology.

Many productivity and utility applications that are commonly used on smartphones hold private data and require authentication to access that data. Some common examples include applications for eReaders, email, notes, and social networks. In most cases, these applications require authentication, but only single factor authentication (a password). The security of these applications could be improved by using a multi-factor authentication system, and given the security and convenience of voiceprint authentication; Voiceprint Vault would provide an excellent solution for many applications that fall under these categories.

Voiceprint Vault offers synching of user data across the network as a built-in feature. Many file synching services are becoming available to smartphone users within the last few years, as data and services migrate to the cloud. A similar synching service could easily be built for Android that simply extends the behavior that is built-in to the Voiceprint Vault framework.

For corporate environments, there is a real need to control access to intellectual property among employees, and to also allow employees to collaborate with one another on projects. Voiceprint Vault is well designed to be extended to handle this use case in the future. The core functionality is already present, but a more advanced permission scheme would be required to allow multiple users to share documents. A corporation that desired to implement a system like this could even run an extension of Voiceprint Vault on their own server with some modifications and control access to files across many registered users. This would require more significant developments to the Voiceprint Vault framework but could provide significant advantages to corporations dedicated to security and collaboration among employees.

Chapter 8: Conclusion

8.1 CONTRIBUTIONS

The rapid move toward smartphones, mobile applications, and cloud-based services requires that authentication systems adapt to be more secure. At the same time, consumer demand and market pressures require the authentication systems to be as convenient as possible to users. The Voiceprint Vault system provides fast and secure voice authentication to users using trusted DSP algorithms through a web interface. Voiceprint Vault allows application developers to integrate this multi-factor authentication system with new and existing applications.

Under the Voiceprint Vault authentication framework, new users can create an account by providing an email address, password, and voice sample. Following the account creation, users can then access the account by speaking a simple passphrase. Since this system uses text-independent speaker verification, the user can alter their passphrase on each login without any changes on the server, but a consistent passphrase may provide more accurate verification. When users access their account, their voice sample is analyzed and compared to their training recordings.

Voiceprint Vault strives to provide several key benefits to users and developers in order to make this system most effective. The built-in data storage and synchronization through App Engine is one of the chief benefits. This allows users to keep all their data up-to-date across their connected devices. It also eases the application developer's job by saving the trouble of implementing their own data storage. The server uses trusted security models to ensure that user data is protected and communications cannot be intercepted. Application developers can create dedicated repositories for each app that

uses this service, so their user data and voice samples are isolated from other applications.

Voiceprint Vault allows developers of all platforms to integrate this service. All information that is communicated between the client and server is done by way of parameters and files in web forms. The data included does not preclude any platform from taking advantage of the Voiceprint Vault authentication service.

A reference implementation of a Voiceprint Vault application is provided to demonstrate the use of this service. The reference implementation protects access to personal notes, tasks, and dates using voice authentication. A Java library and resource files are also provided to facilitate the use of this service in an Android app. Step-by-step instructions for integrating the Voiceprint Vault framework into a new application are described.

8.2 RELATED WORK

Voice verification services have been implemented by various companies and researchers, but no other services such as this system could be found that provide a conveniently integrated framework with the authentication and data storage concerns handled by a server. Selected products in the voice processing field are discussed in this section.

8.2.1 Speech Recognition Products

This technology is often compared to Samsung's S Voice or Apple's Siri. However, these products fill a very different purpose related to voice processing. While Voiceprint Vault verifies a user's identity using a text-independent passphrase, S Voice and Siri exist specifically to interpret and react to words spoken by the user. Speech recognition products are different than speaker verification products in that they are

typically used to relay commands or questions to a software service, whereas speaker verification is intended to verify a user's claim that they are who they claim to be. [SIR13] [SAM13]

Voiceprint Vault completely disregards what words are spoken so that users have the convenience of speaking any passphrase. A text-dependent verification system can be more cumbersome because it requires users to remember their exact passphrase and keep it secret from others. Voiceprint Vault does not place either of these requirements on users.

8.2.2 Dedicated Speaker Verification Systems

Vanguard Financial Institution offers a typical example of a voice verification service. Vanguard uses speaker verification to confirm the identity of callers to their automated banking line. Vanguard's authentication system is very similar to the Voiceprint Vault system. A customer is required to provide a voice sample when they set up voice verification. On future calls, they are asked to repeat the phrase "at Vanguard, my voice is my password," in order to proceed with their transaction. This helps to ensure secure transactions for Vanguard and remains convenient for the customer. [VAN11]

While this system is very similar to the Voiceprint Vault system, the primary difference is scalability. The Vanguard system is set up for Vanguard's sole use and is a part of their larger security plan. Voiceprint Vault seeks to remove the need for client applications to maintain any authentication protocol of their own other than plugging into the Voiceprint Vault framework. Voiceprint Vault also allows vast numbers of clients to use the same framework, rather than requiring that each client application implement their own authentication server.

8.2.3 Generic Speaker Verification Software

Several vendors of biometric software sell voice recognition software that provides a similar authentication system to Voiceprint Vault. Recognition Technologies, Inc. and CSID are two examples of organizations with voice recognition products. These products are similar to the voice authentication system described at Vanguard, but are generic products that can be customized and deployed at many locations. While this is closer to the service that Voiceprint Vault provides, it still requires that clients use the software to implement their own authentication system. [CSI13] [REC13]

For some companies, it is desirable to maintain their own authentication system, which may be much broader and include more features than the Voiceprint Vault system. However, for clients with minimal authentication and data storage needs, Voiceprint Vault maintains a strong advantage over this paradigm when it comes to scalability and ease of integration for client applications. Voiceprint Vault can easily be integrated with many diverse client applications and does not require that the client applications implement their own authentication system.

8.3 EXTENSIONS TO THIS APPLICATION

There are many extensions to this application that can be made to improve the reliability of the speaker verification, ease integration for client application developers, and include tighter security measures around the framework. Some of these extensions are discussed in this section.

8.3.1 Local Encryption for all User Data

This framework provides many security measures to protect users' private data. However, the ultimate standard in securing user data would be to include local encryption

for the data that could not be decrypted by the server. A system that uses this feature could be set up as follows.

Users taking advantage of this feature would be required to use a password along with their voice authentication. The user would enter their email address and upload their verification recording when logging into the application. Upon successful authentication of their voice, the user would be granted access to the application. After they have saved any data that they want within the application, the user would be required to supply their password. This password would never be sent to the server, but instead a secure hash of the password would be computed locally. The password has becomes the encryption key. The encryption key is then used to encrypt the user's entire block of data.

That data block along with a hash of the encryption key would be stored in the Datastore. When the user authenticates again with their voice and email, the encryption key hash and data block would be downloaded from the Datastore (still as an encrypted data block). The user would be prompted for their password, which would once again be hashed by the application to derive the decryption key (which should be identical to the encryption key). After verifying that a hash of the decryption key matches the downloaded encryption key has, the decryption key would be used to decrypt the data block containing their existing user data. After the user has modified the data as desired, it is again encrypted and uploaded to the Datastore.

This system uses local encryption of the data only, and the encryption key is never shared with App Engine, nor is the password. This means that even a rogue App Engine developer could not view the contents of the user's private data. This system is slightly more dangerous to the user because if they ever forget their password, there is no possible way to recover their data, but it represents the ultimate in security.

8.3.2 Notification of Access from New Devices

Another feature that could be implemented to better protect users of this application is a notification message that is emailed to the user every time their account is authenticated from a new device. This would allow users to realize if their account has been penetrated and they could take reactive measures to secure any compromised data. This feature could potentially even be extended to require that when authenticating from a new device, the user must enter a one-time code that is emailed to them. This would require that a user logging in under a given email address actually does have access to that email address.

8.3.3 Group Access to Shared Data

This feature would allow multiple users of a client application to share and modify common data. This could be an important feature for corporations or teams that want to collaborate on a secure document. The current framework only supports user application data owned by individual users, but this framework could be extended with a list of authorized users and permission levels for shared data. This collaborative feature would extend the functionality and flexibility of Voiceprint Vault while maintaining a strong security model for accessing private data with voice authentication.

Glossary

Speech Recognition: The process of extracting the words spoken from an audio sample

Speaker Recognition: See *Speaker Identification*

Multifactor Authentication: The process of authenticating a user based on factors that fall under at least two of three possible categories (knowledge, possession, and inherence)

Biometric Factor: See *Inherence Factor*

Inherence Factor: A piece of information extracted from a person's inherent physical or biological characteristics (fingerprint, retinal scan, voiceprint, etc.)

Voiceprint: An individual's vocal features that uniquely identify their voice

Mel Frequency Cepstrum Coefficients (MFCC): Coefficients that represent a logarithmic weighting of the frequency domain for an audio sample

Speaker Identification: The process of identifying what user from a group of registered users is speaking

Speaker Verification: The process of verifying that a speaker is who they claim to be based on prerecorded training samples

Text-Independent Algorithms: Voice-processing algorithms that do not require the user to speak a pre-determined word or phrase

Text-Dependent Algorithms: Voice-processing algorithms that require the user to speak a pre-determined word or phrase

Advanced Encryption Standard (AES): Symmetric key encryption algorithm that renders brute force attacks computationally infeasible

PBKDF2: A key derivation function that applies a cryptographic hash along with a salt value to generate an irreversible password hash

HTTPS: The HTTP Secure protocol provides for secure end-to-end communication over an insecure path

Acceptance Threshold: The value which all feature comparison differences must fall below in order for the user to be successfully verified

References

[YFN13] Five Star Equities. "Number of Smartphones Around the World Top 1 Billion." *Yahoo! Finance*, 19 Oct. 2012. Web. 12 Mar. 2013.

[SIR13] Apple, Inc. “iOS: Siri.” http://www.apple.com/ios/siri. 15 Mar. 2013.

[RAS04] Hasan, Rashidul, et al. Speaker Identification Using Mel Frequency Cepstral Coefficients. Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology. 28 Dec. 2004.

[KAM10] Kamruzzaman, S. M. et al. Speaker Identification using MFCC-Domain Support Vector Machine. Department of Computer Science and Engineering University of Rajshahi. 2010.

[ZHE01] Zheng, Fang, Guoliang Zhang, and Zhanjiang Song. Comparison of Different Implementations of MFCC. Center of Speech Technology, Tsinghua University, Beijing. Sept. 2001.

[LYO09] Lyons, James. "Mel Frequency Cepstral Coefficient (MFCC) tutorial." *Practical Cryptography*, 2009. Web. 20 Apr. 2013.

[SUK12] Sukor, Abdul Syafiq. Speaker Identification System Using MFCC Procedure and Noise Reduction Method. University Tun Hussein Onn Malaysia. Jan. 2012.

[SCH11] Schedl, Markus. CoMIRVA: Collection of Music Information Retrieval and Visualization Applications. 19 Dec. 2011. Web. 18 Oct. 2012.

[GDS12] Google, Inc. Using the Datastore. *Google Developers*. 19 June 2013. Web. 05 Nov. 2012.

[GRE10] Greensted, Andrew. Java Wav File IO. *The Lab Book Pages*. 25 Sept. 2010. Web. 08 Nov. 2012.

[DOM13] Do, Minh. DSP Mini-Project: An Automatic Speaker Recognition System. University of Illinois at Urbana-Champaign. Web. 14 May 2013.

[ORA13] Oracle. Java Servlet Technology Overview. 05 Mar 2013.

[GAE13] Google Developers. App Engine Java Runtime Environment. 25 June 2013. 5 July 2013.

[JCP13] Oracle. Javax Crypto Package Documentation Summary. 15 Nov. 2013.

[COM13] Flosi, Stephanie. ComScore. May 2013 U.S. Smartphone Subscriber Market Share Report. 28 June 2013.

[VAN11] Vanguard. Voice Verficiation Secure Account Access. Feb 2011. https://personal.vanguard.com/pdf/c106.pdf

[SAM13] Samsung Electronics Company. Samsung Galaxy SIII S Voice. http://www.samsung.com/global/galaxys3/svoice.html. 5 July 2013.

[CSI13] CSID. Voice Biometrics Identity Protection. 05 July 2013.

[REC13] Recognition Technologies, Inc. Voice Recognition Products. 05 July 2013.

[AEN13] Google Developers. Google App Engine Features. 11 June 2013.

[MCC07] McCarthy, Caroline. CNET News. “Study: Identity theft keeps climbing”. 06 March 2007.

Vita

Marty Henderson was born and raised in Kentucky. He attended Vanderbilt University in Nashville, TN and majored in Computer Engineering and Mathematics. After graduating from Vanderbilt in 2007, he moved to Austin, TX to begin working for National Instruments as an applications engineer. He remains employed at National Instruments as an installer developer. He enrolled in the University of Texas Software Engineering program in fall of 2011.

Permanent email: Paul.Henderson.Voiceprint@gmail.com

This report was typed by Paul Martin Henderson.